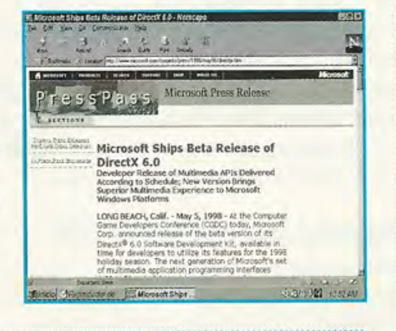


Dreamweaver 1.2

Ya está disponible la nueva versión de Dreamweaver, la herramienta visual líder para el diseño profesional de webs.
Dreamweaver 1.2 ofrece a los desarrolladores web la posibilidad de distribuir impactantes páginas web a través de conexiones de Internet con bajos anchos de banda para beneficiarse de las ventajosas características que ofrece Dynamic HTML para los usuarios de navegadores 4.0, pero manteniendo, al mismo tiempo, la compatibilidad con los navegadores 3.0. Esta nueva versión incorpora varias innovaciones que mejoran la productividad del diseñador web y facilitan su uso.
Nuevas características de Dreamweaver 1.2
Compatibilidad con los navegadores 4.0 y 3.0: Ahora los diseñadores de webs pueden crear páginas que incorporen las ventajas de Dynamic HTML tales como las capas y las hojas de estilo enlazadas para usuarios de navegadores 4.0 asegurando, al mismo tiempo, la compatibilidad con el navegador 3.0.
Accionando el comando convertir a navegadores 3, la página se hace automáticamente compatible para los navegadores 3.0 al mismo tiempo que permite, a los usuarios de navegadores recientes, disfrutar de Dynamic HTML. Dreamweaver 1.2 también incluye un comportamiento que, automáticamente, envía al usuario a la página optimizada en función de su navegador.

Directx 6.0



Microsoft ha presentado la primeras versiones de las DirectX 6.0. Se supone que corregirán todos los fallos de versiones anteriores y aprovecharán al máximo la potencia de los nuevos ordenadores. Toda la información posible se encuentra disponible en la web de Microsoft, y muy pronto, muchos juegos se construirán con las DirectX 6.0

Forum Filemaker Pro

El viernes 26 de junio se celebrará en Madrid el primer Forum FileMaker Pro. La meta de este Forum es dar a conocer la nueva orientación de FileMaker Inc. después de la disolución de Claris y mostrar, a todos los desarrolladores, los programas FileMaker Pro 4.0, FileMaker Server y Home Page 3.0 a través de la realización de un proyecto global. De esta forma, los organizadores pretender demostrar a la comunidad de desarrolladores, a traves de la presentación del nuevo FileMaker Pro 4.0 Developer Edition, la validez de la plataforma FileMaker para realizar sus proyectos. Además, y para dar un aspecto práctico al evento, se invitará a algunos destacados miembros españoles del FileMaker Software Alliance (FSA) para que enseñen sus productos basados en FileMaker.

Macromedia presenta Fireworks

Macromedia ha lanzado Fireworks, la primera herramienta de producción que aporta un entorno unificado para crear, optimizar y producir gráficos de alta calidad para la web. Pensado para cubrir las necesidades de los diseñadores profesionales de webs, esta nueva herramienta incorpora avanzadas tecnologías tales como la previsualización en la exportación, el completo control sobre la compresión y las paletas de color, la generación automática de los estados de botón y de Java Script 3rollovers2, así como la posibilidad de crear textos y efectos editables en todo momento. Fireworks permite a los diseñadores, crear los gráficos y animaciones más compactos, en el menor número de etapas.

Sumario

Apple y la bolsa

Desde que Apple ha sacado a la venta su nueva serie G3, las acciones de la compañía han ganado en el segundo trimestre 55 millones de dólares en la bolsa de Wall Street. Esperemos que la conocida compañía de la manzana se recupere de sus horas bajas y muy pronto vuelva al lugar que le corresponde.





En nuestro CD de portada incluimos el siguiente material:
- **Las fuentes de código de los ejemplos comentados en 3D manía.**
- **COOL EDIT: Editor de samples para generar efectos de sonido de calidad.**
- **Ejemplos del curso de DIV Games Studio.**

Autor: Jesús de Santos

Ficheros 3DS

Hasta ahora, todos nuestros ejemplos han sido figuras muy simples que tenían como misión mostrar la utilidad de los temas tratados. Según avancemos con temas más complejos necesitaremos figuras algo más complejas.

Ante este problema existen varias soluciones posibles. Sin duda alguna, la más artesanal y costosa es la de realizar un editor propio de escenas y objetos 3D, lo que implicaría, del mismo modo, diseñar un formato propio. En el otro extremo está la opción de usar algún modelador 3D ya existente, utilizando el formato propio que este programa utilice. Nosotros vamos a elegir un punto medio entre ambas opciones; no vamos a diseñar un editor 3D propio por lo costoso que puede llegar a ser esta tarea, más aún cuando ya existen programas muy buenos que están en continua expansión. Con el fin de simplificar un poco las cosas, crearemos nuestro propio formato a partir de un editor externo no creado por nosotros. La idea de usar un formato propio es la de escoger únicamente aquellos datos que necesitemos para nuestras pruebas y programas. De este modo, ahorramos espacio y aislamos cualquier dato que no nos sea necesario.

Entre los muchos programas de diseño 3D, nos hemos decidido por el de la empresa AutoDesk, en concreto, el programa 3DStudio que emplea el formato 3DS para sus ficheros. Las razones de la elección son sencillas: el formato 3DS goza de una gran popularidad dentro del mundillo de los videojuegos y de las demos, por lo que podemos encontrar gran documentación y numerosos ejemplos en muchos lugares de la red. Hay que decir que las últimas versiones de 3DStudio funcionan, exclusivamente, bajo Windows y reciben el nombre de 3DStudioMAX (la última versión es la 2.0). Estas recientes versiones trabajan con un formato mucho superior al 3DS: el formato MAX. Debido a la complejidad de este último, la lectura eficiente de un fichero MAX sólo puede realizarse mediante el SDK que Kinetix distribuye de un modo comercial. Nosotros nos vamos a centrar en el formato 3DS que era el principal de las versiones de DOS del 3DStudio (su última versión es la 4.0). A pesar de que el formato MAX domina últimamente en los programas 3DStudio, siguen existiendo opciones de importación y exportación de 3DS a Max y viceversa (ver figura 1). Por lo tanto, el formato 3DS sigue siendo una buena opción. Los ficheros 3DS son binarios, es decir, no son leíbles desde un editor de textos normal. El propio 3DStudio puede generar ficheros en formato de texto que, después, pueden ser procesados por un correspondiente lector o 'parser'. Estos formatos son el 'ASC' (3DStudio 4.0) y el 'ASE' (a partir de la versión 2.0 del MAX), este último mucho más potente que el primero. Generalmente, en este tipo de ficheros no se suele incluir toda la información posible, sino sólo aquella que pueda parecer útil a los desarrolladores. Además, la realización de un 'parser' suele ser más incómoda que un lector de formato binario. Principalmente por estas razones nosotros vamos a tratar 3DS. Kinetix hizo, hace relativamente poco tiempo, de dominio público la estructura interna de los ficheros 3DS. De todas formas, antes de esto, ya era sobradamente conocido por todos los que nos movemos por este mundo gracias a medios de intercambio como Internet o las news.

ESTRUCTURA INTERNA DE UN 3DS

La estructura de un formato 3DS está jerarquizada gracias a bloques de control denominados 'chunks' (es preferible emplear este término antes que producir más daños al castellano). Un chunk es un bloque de información con la siguiente estructura:

BEGIN CHUNK

ID : 2 bytes
Longitud : 4 bytes

END CHUNK

El campo 'id' nos indica el tipo de información que sigue a continuación del chunk. Por otro lado, el campo 'longitud' indica la longitud total de la información que contiene dicho chunk. Es importante destacar que en este campo 'longitud' va incluido el tamaño del propio chunk, de modo que la longitud mínima de un chunk son seis bytes (longitud de un chunk). A continuación de la cabecera viene la información del chunk, que varía según el campo 'id'. En la información de un chunk podemos encontrar más chunks (sub-chunks del primero) y, a su vez, dentro de un sub-chunk más chunks. De este modo jerárquico se estructura toda la información contenida en el 3DS. Aunque, en principio, el empleo de chunks puede parecer un poco engorroso, tiene una gran ventaja. Si desconocemos el significado de algún chunk podemos saltarlo fácilmente (pues sí conocemos su longitud) y seguir leyendo tranquilamente. Esto favorece del mismo modo a los desarrolladores del formato 3DS que pueden dejar oculta información de un modo muy sencillo.

Pasamos ahora a dar una vista general de los chunks más interesantes. Para que el lector pueda seguir fácilmente la explicación, en la figura 1 aparecen gran parte de los chunks que forman un 3DS. Están estructurados jerárquicamente según la tabulación.

No vamos a detallar aquí uno por uno todos los chunks (además, existen muchos más de los indicados en el listado 1), simplemente tratamos aquellos que vamos a utilizar o aquellos otros que pueden ser interesantes para un uso futuro. Lo que se muestra en el listado 1 es una representación simbólica de los número correspondientes al 'id' de cada chunk. Así, por ejemplo, para el chunk con id: 4D4D empleamos la representación CHUNK_MAIN. Los ejemplos compilables que acompañan al Cd de este mes contienen cabeceras en C con macros definidas para los Chunks.

El chunk principal, CHUNK_MAIN, contiene a todos los sub_chunks del 3DS. Estos se dividen en dos grupos: CHUNK_KEYFRAMER, que contiene toda la información relativa al KeyFramer del 3DStudio, es decir, la animación de los objetos, cámaras y luces mediante interpolación de Splines. Trataremos esta sección muy a fondo en futuros artículos pero, por este mes, nos vamos a centrar en el chunk CHUNK_OBJMESH que es donde se encuentra las información que hace referencia al 3dEditor. Dentro del CHUNK_OBJMESH encontramos dos sub_chunks con importante información: CHUNK_MATERIAL contiene toda la información acerca de los materiales que han

LISTADO 1

```
CHUNK_MAIN
    CHUNK_OBJMESH
        CHUNK_TRIMESH
          CHUNK_VERTLIST
          CHUNK_FACELIST
            CHUNK_FACEMAT
            CHUNK_SMOOTHLIST
            CHUNK_BOXMAP
          CHUNK_MAPLIST
          CHUNK_TRMATRIX
          CHUNK_COLOR
          CHUNK_TEXT_INFO
        CHUNK_LIGHT
          CHUNK_SPOTLIGHT
                    CHUNK_CAMERA
      CHUNK_MATERIAL
        CHUNK_MATNAME
        CHUNK_AMBIENT
        CHUNK_DIFFUSE
        CHUNK_SPECULAR
        CHUNK_TEXTURE
        CHUNK_BUMPMAP
        CHUNK_MAPFILE
    CHUNK_KEYFRAMER
      CHUNK_FRAMES
      CHUNK_OBJINFO
        CHUNK_OBJNAME
        CHUNK_PIVOT
        CHUNK_OBJNUMBER
        CHUNK_TRACKPOS
        CHUNK_TRACKROT
      CHUNK_CAMERAPOS
        CHUNK_TRACKFOV
        CHUNK_TRACKROLL
      CHUNK_CAMERATARGET
```

sido introducidos en el programa con el 'material editor' del 3DStudio. De momento, nosotros no vamos a leer texturas, simplemente queremos mostrar un objeto en alámbrico. De todas formas, también trataremos este chunk en futuros artículos. El otro sub_chunk que nos interesa es CHUNK_OBJBLOCK donde se definen los diferentes objetos de la escena: luces (CHUNK_LIGHT), cámaras (CHUNK_CAMERA) y mallas de triángulos (CHUNK_TRIMESH). Es, en este último chunk, donde centraremos principalmente este artículo. Este chunk y todos sus sub_chunk contienen la información necesaria sobre la representación mediante triángulos de las mallas que podemos editar en el 3DStudio.

EL CHUNK_TRIMESH

Este chunk, el CHUNK_TRIMESH, se subdivide, a su vez, en otros chunks de los cuales nos van a interesar los siguientes:

• *Chunk_Vertlist*

En este chunk se almacenan todos los vértices que forman el objeto. Vienen dados en coordenadas de mundo así que no es necesaria ninguna transformación. Los primeros 2 bytes del chunk nos informan del número total de vértices. A continuación, aparecen todos los vértices seguidos en formato de coma flotante con precisión simple con la siguiente estructura:

```
BEGIN VRT

        x : float (4 bytes)
        y : float (4 bytes)
        z : float (4 bytes)

END VRT
```

• *Chunk_Facelist*

Información acerca de las caras que forma el objeto. Las caras son triángulos y contienen índices a la lista de vértices anterior (pero esto no quiere decir que la lista de vértices tenga que aparecer antes que la de caras en el fichero 3ds). Como en el chunk de vértices, la primera información que encontramos son 2 bytes que nos indican el número de caras totales que forman el objeto. Después se indica la información de todas las caras con la siguiente estructura:

```
BEGIN FACE

  pt[0]  : 2 bytes - Vértice 0 (índice a lista de caras)
  pt[1]  : 2 bytes - Vértice 1          ""
  pt[2]  : 2 bytes - Vértice 2          ""
  flag   : 2 bytes - Atributos

END FACE
```

El campo flag contiene información que nosotros no vamos a emplear, ya que detalla todos los aspectos acerca de la visibilidad de las aristas de las caras.

• *Chunk_Maplist*

Este chunk nos informa sobre las coordenadas de textura de los vértices del objeto. Puede que este chunk no aparezca (si no se ha aplicado el 'mapping' en el 3DStudio) en el fichero 3DS. Si aparecen coordenadas de textura entonces existen tantas como vértices del objeto. Las coordenadas de textura se aplican a los vértices en el orden en que aparecían en el chunk de vértices. Una de las grandes desventajas que tiene el formato 3DS es que el mapping viene dado para cada vértice. De modo que si un mismo punto tiene que recibir distinto mapping, el 3DS lo duplica internamente. El formato MAX ya no tiene este problema y almacena las coordenadas de texturas por caras en vez de por vértices. De momento, tampoco vamos a emplear este chunk, pues no vamos a mostrar objetos texturados (pero lo haremos, por supuesto). Indicar, simplemente, que cada mapping



FIGURA 2. CON EL EJEMPLO DE ESTE MES PODREMOS CARGAR CUALQUIER FICHERO 3DS Y MOVERNOS LIBREMENTE POR EL CON NUESTRO ENGINE.

consta de dos componentes (u y v) expresados en flotante simple (float).

• *Chunk_Trmatrix*

Contiene información sobre la matriz del objeto. Se almacena como 12 números flotantes formando una matriz 4x3. La última columna siempre es <0.0, 0.0, 0.0, 1.0 > y, por tanto, no se almacena. La matriz no tiene por qué ser ortogonal (ver artículo sobre matrices y transformaciones locales en el número 3 de esta revista). De momento, no vamos a usar este chunk. Cuando leamos varios objetos con

LISTADO 2

```
#define NAME_SIZE 20

typedef struct {

   float x,y,z;              // <x,y,z>

}G3D_VECTOR;

typedef struct {

   float a,b,c,d;            // Ax+By+Cz+D=0

}G3D_PLANE;

typedef struct {

   DWORD pt[3];
   G3D_PLANE eq;

}G3D_FACE;

typedef struct {

   STRING name[NAME_SIZE];

   DWORD no_vrt;
   G3D_VECTOR *vrt;

   DWORD no_faces;
   G3D_FACE *faces;

}G3D_FILE;
```

LISTADO 3

```
void render(GFXMODE_EX *gfx, matrix &local, camera &cam, OBJECT *obj)
{

        [.......]

// Transformar los planos de la pirámide de visión a coordenadas
// locales. Es necesario para hacer clipping 3d en coordenadas
// locales

        [.......]

// Transformación y proyección de los vértices que se deban ver

  for(i=0;i<obj->no_vrt;i++) {

// Inicialmente el vértice entra en pantalla

     (obj->vrt+i)->flag=0;
     cur_vrt=&(obj->vrt+i)->local;

// Clipping con cada uno de los planos de visión

     for(k=0,c=1;k<CLIP_PLANES;k++,c<<=1) {

        if(cur_vrt->x * local_frustum[k].a +
           cur_vrt->y * local_frustum[k].b +
           cur_vrt->z * local_frustum[k].c < -local_frustum[k].d)

              (obj->vrt+i)->flag|=c;

     }

// Si el vértice entra dentro del cono de visión tranformar y
//  proyectar

     if((obj->vrt+i)->flag==0) {

        temp=local_2_view.transform((obj->vrt+i)->local);

                invz=1/temp.z;
                (obj-
>vrt+i)>scr.x=cam.centerx+cam.xscale*temp.x*invz;
                (obj->vrt+i)-
>scr.y=cam.centery+cam.yscale*temp.y*invz;

                [........]

     }

  }

        for(i=0;i<obj->no_faces;i++) {

// BACK FACE :
// Comprobamos si la cámara está delante o detrás de este polígono.

        [........]

        triangle[0].local=(obj->vrt+(obj->faces+i)->pt[0])->local;
        triangle[0].flags=(obj->vrt+(obj->faces+i)->pt[0])->flag;
        triangle[0].owner=&(obj->vrt+(obj->faces+i)->pt[0])->scr;

        triangle[1].local=(obj->vrt+(obj->faces+i)->pt[1])->local;
        triangle[1].flags=(obj->vrt+(obj->faces+i)->pt[1])->flag;
        triangle[1].owner=&(obj->vrt+(obj->faces+i)->pt[1])->scr;

        triangle[2].local=(obj->vrt+(obj->faces+i)->pt[2])->local;
        triangle[2].flags=(obj->vrt+(obj->faces+i)->pt[2])->flag;
        triangle[2].owner=&(obj->vrt+(obj->faces+i)->pt[2])->scr;

        clip_poly[0]=&triangle[0];
        clip_poly[1]=&triangle[1];
        clip_poly[2]=&triangle[2];

// Clipping 3d. La función devuelve el número de vértices finales del
//          polígono final.

           na=clip3d(clip_poly,&final,local_frustum);

// Proyección de los vértices finales

        for(j=0;j<na;j++) {

// Si el vértice ya fue transformado y proyectado entonces reusar la
//          informacion

           if(final[j]->owner!=NULL) {

              out[j].x=(final[j]->owner)->x;
              out[j].y=(final[j]->owner)->y;

           }

           else {

              temp=local_2_view.transform(final[j]->local);

                 invz=1/temp.z;
              out[j].x=cam.centerx+cam.xscale*temp.x*invz;
                 out[j].y=cam.centery+cam.yscale*temp.y*invz;

// De nuevo clamp

                 [........]

           }

        }

// Pintado en alámbrico.

                 [........]

           }
         }
```

coordenadas locales en cada uno, este apartado nos será de gran utilidad.
Antes de finalizar el apartado, unas pequeñas observaciones sobre el formato max de 3DStudio Max. Este formato es de una complejidad mucho mayor que el 3ds. En MAX, gran parte de los objetos son almacenados como fórmulas y triangularizados en tiempo de carga del objeto. Pongamos un ejemplo: en el 3DS una esfera se almacena con todos su vértices y caras. En cambio, en un fichero max se guardaría la posición de la esfera, su radio y su resolución. Cuando cargamos el fichero en el programa es el propio 3DStudio el que se encarga de generar los triángulos que corresponden a la esfera. Con esto queremos decir, que la lectura en 'bruto' de un max es prácticamente imposible. Destacar también que los conversores de 3DS a max, y viceversa, que vienen con el 3DStudio MAX tienen bastantes fallos. Así, si queremos pasar de un fichero max a 3DS podemos encontrar problemas con la colocación de la textura y con la animación del keyframes. En numerosas ocasiones toda esta información se almacena incorrectamente. El paso de 3DS a max suele causar muchos menos problemas funcionando, casi siempre, correctamente.

De momento, con obtener una visión 3d del objeto en alámbrico nos podemos dar por satisfechos

Nuestro formato propio: G3D

Una vez conocida por encima la estructura de los ficheros 3DS es necesario definir nuestro propio formato. Éste, irá creciendo cada mes para incluir todo aquello que vayamos necesitando. Es mucho más sencillo iniciarse con un formato sencillo que empezar a complicar las cosas con información que, de momento, no vamos a emplear. Básicamente lo que nosotros queremos es un formato que guarde información acerca de los vértices y las caras del objeto. De momento, con obtener una visión 3d del objeto en alámbrico nos podemos dar por satisfechos. Vamos a ello. En el listado 2 aparecen todos los tipos que van a definir nuestro formato que vamos a 'bautizar' con la extensión G3D.
Nuestro formato G3D es muy sencillo: básicamente se compone de una lista de vértices y una lista de caras con índices a los vértices. Podemos obtener estos datos de un modo bastante directo del 3DS. El único trabajo que tenemos que realizar es ir uniendo todos los objetos que el 3DS nos da en uno único. Para realizar este paso tenemos que reajustar correctamente los índices de las caras a la nueva lista global de vértices. Si queremos emplear todos los algoritmos que hemos ido describiendo los últimos meses necesitamos también guardar la ecuación del plano de cada cara. El formato 3DS no nos la da directamente, pero podemos calcularla muy fácilmente como ya hemos explicado en un artículo anterior sobre ocultación de caras.
Para recorrer el fichero 3ds a lo largo de sus chunks implementamos una función auxiliar que avanza por los chunks conocidos y pasa 'transparentemente' por los desconocidos. La llamamos next_chunk(). El recorrido que tenemos que hacer por el 3DS es el siguiente:

1. **Saltamos CHUNK_MAIN, CHUNK_OBJMESH, CHUNK_MATERIAL con tres llamadas a next_chunk**
2. **Buscamos el primer CHUNK_OBJBLOCK. Esto, generalmente, consiste en saltarnos toda la lista de materiales que preceden a la definición de los objetos.**
3. **Dentro de OBJBLOCK buscamos los chunks CHUNK_TRIMESH que contienen la información que nos interesa de los objetos.**
4. **Leemos la información de cada uno de los objetos y la vamos almacenando en un objeto global modificando correctamente los índices de las caras a los vértices. Calculamos la ecuación de plano de cada uno de los triángulos y la almacenamos de forma correcta.**
5. **Escribir toda la información a un fichero g3d.**

En el primer ejemplo que viene en el CD de la revista se incluye un completo conversor de 3ds a g3d. Vamos a usar constantemente este formato (y lo vamos a ir mejorando de forma progresiva) a partir de este artículo. Y para no esperar más, vamos con nuestro primero uso de un fichero g3d!

NUESTRO PRIMER USO DEL FORMATO G3D

No vamos a esperar ni un momento más para usar el formato que acabamos de definir. Para ello usaremos una variante mejorada del ejemplo que apareció en el último artículo sobre clipping 3d. En dicho artículo empleábamos una figura muy simple definida prácticamente a mano y «metida» dentro de los datos del programa. Ahora podemos comprobar todo el potencial del clipping 3d con escenas mucho más complejas y movernos por mundos más realistas que hayan sido construidos con programas de diseño 3D, tales como LightWave, 3DStudio, 3DStudioMAX y cualquier formato, siempre que dispongamos de la utilidad correspondiente para transformar a 3DS y luego con el programa que nosotros hemos diseñado convertir a g3d.
Hemos realizado una importante mejora con respecto al código de clipping 3d del mes pasado. Nuestros objetos 3D están estructurados ahora de tal manera que los vértices son compartidos por distintas caras. Si recortáramos, transformáramos y proyectáramos los vértices por cada una de las caras estaríamos repitiendo muchos cálculos. Hemos de realizar los cálculos una vez por cada vértice y luego que cada cara acceda a valores calculados para sus vértices. De esta forma, los cálculos sólo se realizan una vez por cada vértice.
Esta nueva mejora introducida al algoritmo de clipping 3d se realiza con los dos siguientes cambios:

- Ahora antes de recorrer la lista de caras, recorremos la lista de vértices y los clasificamos con respecto a los planos de la pirámide de visión. Si entran en pantalla son transformados y proyectados para su posterior uso.
- Redefinimos nuestra estructura TCLIP con un nuevo campo *'owner'*

```
typedef struct {

        VECTOR local;
        BYTE flags;
        PIXEL *owner;

}TCLIP;
```

Este campo inicialmente es un puntero a la estructura PIXEL (coordenadas de pantalla <x,y>) que contiene al vértice proyectado en pantalla. Debido a que el clipping 3d puede introducir nuevos vértices, éstos se diferencian de los fijos porque su campo se marca con NULL, (es decir, sin dueño). Una vez acabado el clipping 3d aquellos vértices sin dueños tienen que ser transformados y proyectados, pues han sido creados por el procedimiento de clipping. En el listado 3 tenemos los principales cambios que ha sufrido el procedimiento de pintado de objetos en pantalla. Con esta pequeña modificación hemos conseguido acelerar un poco más nuestro motor 3d. Iremos viendo más técnicas y trucos para conseguir arañar al máximo la suavidad.
Con todo esto podemos finalizar el artículo. En nuestro afán por conseguir escenas 3D más realistas, el próximo mes empezaremos a pintar objetos 'sólidos'. Veremos los problemas que esto conlleva y distintos algoritmos para conseguir toda la velocidad que nuestros juegos y demos necesitan.

Jesús de Santos García
icg_ent@hotmail.com

Autor: Antonio Marchal

Curso de programación Avanzada con DIV Games Studio

Este mes haremos una especie de mezcla con pequeñas partes de DIV que no tienen mucho que ver unas con otras, ya que lo único en lo que se parecen es que son temas avanzados del lenguaje. Algunos de éstos han sido requeridos por lectores, normalmente vía e-mail, por ello esperamos que sean correspondidas sus peticiones.

Hay que tener en cuenta que algunas partes del lenguaje son más complicadas que otras. Esto ocurre cuando se quiere escribir en el disco duro, cuando se usa la sentencia CLONE, o bien cualquier otra operación que, para un juego, normalmente no se tiene que usar. Otro de los temas que vamos a tratar es el de la utilización de trucos de programación que nos facilitarán nuestra tarea a la hora de realizar un videojuego. Y si queda espacio, iremos más allá, resolviendo las dudas que se nos hayan quedado en el tintero.

MANOS A LA OBRA

Empezaremos hablando de cuestiones que únicamente usamos en algunos juegos, por ejemplo, el ratón; es decir, si vamos a desarrollar un juego de naves, no lo necesitaremos, pero si, por el contrario, creamos un juego de estrategia, sí.

Una cuestión a reseñar es el aspecto gráfico, para el que disponemos de varias variables dentro de la estructura

Aquellos que habéis leído el manual, o la ayuda, sabréis que existe una estructura predefinida, denominada *mouse*, en la que podemos leer y escribir, para posicionar o tomar los datos, respectivamente, del ratón en un momento dado. La estructura es la siguiente:

```
STRUCT mouse;
x;          // Coordenada X del ratón en pantalla
y;          // Coordenada Y del ratón en pantalla
z;          // Prioridad de impresión del ratón
graph;      // Gráfico del ratón a utilizar
file;       // Fichero que contiene el gráfico a utilizar
angle;      // Angulo del gráfico del ratón
size;       // Tamaño del gráfico del ratón
flags;      // Distintas banderas que despejan el gráfico y le hacen
            trasparente
right;      // Botón derecho del ratón
middle;     // Botón central del ratón
left;       // Botón izquierdo del ratón
END
```

La utilización de esta estructura es bien sencilla; es decir, se usa el nombre de la estructura y el de la variable conjuntamente y separadas por un punto, por ejemplo, *mouse.right*, para manejar el botón derecho del ratón. En cualquier momento podemos leer las coordenadas, o los botones, del ratón y actuar en consecuencia, aparte de poder posicionar el ratón, escribiendo valores en la estructura. En algunos ordenadores, el botón que está situado en el medio no tiene ninguna utilidad debido a los drivers de ratón cargados en memoria.

Una cuestión a reseñar es el aspecto gráfico, para el que disponemos de varias variables dentro de la estructura. Estas variables son semejantes a las que utiliza el resto de procesos, pudiendo elegir el gráfico, aparte de realizar otra serie de movimientos tales como rotarlo, cambiarle el tamaño, hacerlo transparente, etc. Pero algo que tenemos que tener siempre presente es que, cuando leemos el ratón, éste únicamente nos devuelve un punto, que es donde estará situado el punto de control número 0 del gráfico. Al crear el gráfico, debemos tener en cuenta dónde situamos dicho punto de control, al igual que cuando realicemos cualquier operación de rotado o espejado.

Hasta aquí todo muy sencillo, y sin problemas, pero a veces es conveniente que un proceso detecte al ratón, es decir, que compruebe si el cursor del ratón está encima de él o no. Para conseguir esto, el ratón debe tener un gráfico asignado, utilizando, en este caso, la función *collision*, como lo haríamos con cualquier otro proceso, pero usando como tipo de proceso el nombre de la estructura *mouse*. La línea condicional resultante podría quedar como *IF (collision (TYPE mouse)) ...END* y luego, actuar en consecuencia. Hay que señalar que, en esta última condición, solamente se detecta el punto de control número 0, lo mismo que ocurría en el codo de las coordenadas.

Otro problema que surge es el de las regiones de pantalla y el ratón. Estos dos elementos no tienen ninguna conexión, pero como las regiones de pantalla son cuadradas podemos crear una tabla global que guarde los valores de estas regiones. Posteriormente, tendremos la posibilidad de crear un proceso que compruebe si el ratón está dentro de alguna de las regiones de esta nueva tabla, si bien el crear un proceso aparte, que maneje el ratón, su posición, su estado, etc, es algo que podemos utilizar con otros fines. Por ejemplo, podemos elaborar un

proceso que vaya cambiando el aspecto gráfico del ratón, animándolo según queramos nosotros.
En los primeros artículos de esta sección, cuando hablamos de los mapas de durezas se vieron otras técnicas de utilización del ratón, de ahí que remitamos a los mismos para analizar diversos trucos de uso de ratón, así como ejemplos y referencias.

LA OVEJITA DOLLY

Una de las secuencias más extrañas de DIV Games Studios es CLONE. Esta instrucción sirve, como su nombre indica, para clonar un proceso, es decir, crea otro idéntico al proceso donde aparece la sentencia. Pero, asimismo, se le pueden dar otras características, ya que, de lo contrario, la sentencia no tendría una utilidad real. Por ello, podemos observar el modo de funcionamiento dentro del juego ALIEN SUPRIMER, que viene de ejemplo con el entorno. No es muy recomendable su uso, pues debemos tener mucho cuidado con el clon que se crea nuevo. Por este motivo, recomendamos hacer uso de este tipo de procesos únicamente en marcadores y otros procedimientos cuyas diferencias entre unos y otros sean tan simples que no necesiten crear un nuevo proceso completo para su uso.
También es posible que creemos un proceso, luego hagamos un clon, y eliminemos ese proceso, mientras el clon continúa existiendo, por lo que habrá que tener precaución con su utilización. Es más recomendable tener los procesos por separado, siempre que sea necesario varios y muy parecidos, y sólo hacer uso de esta sentencia cuando sea estrictamente necesario, si bien, en condiciones normales no será necesario.
Pero veamos un ejemplo de la utilización de esta sentencia; imaginemos que tenemos un proceso que dibuja el gráfico de un cuadrado en pantalla, haciendo las veces de marcador. Su código podría ser éste:

```
PROCESS marcador()
BEGIN
x=320;
y=200;
graph=4;
LOOP
FRAME;
END
END
```

Si quisiéramos a la vez, crear otro marcador, usando la sentencia CLONE, el código podría ser, en este caso, algo así:

```
PROCESS marcador()
BEGIN
x=320;
y=200;
graph=4;
CLONE
x=160;
y=100;
graph=4;
END
LOOP
FRAME;
END
END
```

Con esto conseguiríamos tener dos gráficos distintos en diferentes coordenadas, pero en un solo proceso. A la hora de detectar colisiones, e interactuar con otros procesos, la cosa se puede complicar, ya que, aunque se detectan cada uno por su lado y se pueden eliminar, no sabríamos exactamente cuál de los

Funciones y estructuras vistas en el artículo

Estructura setup
Guarda valores que podrían ser denominados de bajo nivel, sobre la tarjeta de sonido y sus volúmenes en general. Todos los valores de esta estructura se autoinicializan al cargar DIV Games Studio, por lo que su modificación sólo queda para casos especiales. Las variables incluidas en esta estructura son las siguientes:

card: Determina el tipo de tarjeta de sonido que se posee en el equipo, es decir, la marca y la versión de la misma.
port: Indica el puerto por dónde mandará y recibirá datos la tarjeta de sonido, normalmente el valor por defecto es 220h.
irq: Este valor indica la interrupción que usará la tarjeta de sonido para trabajar; por defecto, en casi todas las tarjetas este valor es 07.
dma: Las tarjetas de sonido suelen usar una canal directo para trabajar con la memoria, que recibe el nombre de dma. Este valor guarda el canal directo que usaremos.
dma2: Algunas tarjetas necesitan dos canales de acceso directo a memoria para trabajar; si se diera el caso, usaríamos este campo para almacenar dicho valor.
master: Controla el volumen general de la tarjeta de sonido; si modificamos este valor, cualquier sonido que esté haciendo el ordenador variará.
sound_fx: Otro volumen, esta vez el que maneja los samples o muestras de sonido. Se distingue del anterior en que éste modifica un tipo de sonido en particular, y el otro en general.
cd_audio: Y, por último, otro volumen, pero esta vez maneja el CD; variando el valor, conseguiremos que las pistas de audio suenen más o menos con un volumen alto.

Funciones
load_pcm(<nombre del archivo>, <ciclico>)
Esta función sirve para cargar un sample o una muestra de sonido en memoria; para ello, debemos indicar el nombre del sonido en el disco duro y si éste va a ser cíclico o no. Nos devolverá un código identificador, con el que podremos hacer reseñas al sonido cargado.
unload_pcm(<codigo del sonido>)
Descarga de memoria el sonido cargado anteriormente; para conseguirlo, debemos incluir como parámetro el código del sonido a descargar. Existe la posibilidad de descargar todos los sonidos a la vez, con el uso de la constante all_text.
sound(<codigo de sonido>, <volumen>, <frecuencia>)
Cuando queramos que un sonido cargado en memoria se ejecute, es decir, suene, debemos usar esta función. Hay que indicar el código del sonido, así como un volumen y una frecuencia, que será con la que suene dicho sonido. La función devuelve un valor, que es el código del canal, por donde está sonando nuestro son.
change_sound(<canal>,<volumen>,<frecuencia>)
Si queremos cambiar el volumen o la frecuencia de un sonido que se esté ejecutando en ese momento, únicamente debemos indicar los nuevos valores, así como el número de canal, al usar esta función. Esta cualidad, la de permitir cambiar volúmenes y frecuencia, es muy útil para sonido cíclicos.
stop_sound(<canal>)
A veces resulta idóneo, sobre todo con sonidos cíclicos, que éstos dejen de ejecutarse, es decir, de sonar, para conseguir "callar" a cualquier canal; para ello, tan sólo debemos indicar el mismo al usar esta función.
play_cd(<numero de pista>,<modo>)
Existe la posibilidad de hacer sonar pistas de audio del CD que llevan incorporado casi todos los ordenadores. Para eso, tendremos que utilizar esta función, cuyos parámetros son el número de pista a ejecutar y el modo, que varía entre continuar sonando, o parar, cuando la pista termine.
stop_cd()
Al igual que existe una función para ejecutar pistas de CD, existe esta otra para detener dicha ejecución. Es decir, que el CD deja de sonar en el momento en que vayamos a usar esta función. No necesita parámetros ya que detiene la ejecución del CD de forma automática donde esté.
is_playing_cd()
Y dentro del apartado CD tenemos esta función que nos permite averiguar si la unidad de CD está sonando en un momento dado o, por el contrario, está parada. Dicha función no necesita parámetros, pero sí nos devuelve un valor indicándonos el estado del CD.
set_volumen()
Si modificamos la estructura setup, exactamente las variables que hacen referencia a los volúmenes, éstos no tendrán efecto hasta que usemos esta función, que sirve para actualizar dichos valores y tenerlos en cuenta.
reset_sound()
Si, por casualidad, debemos modificar los parámetros internos de la tarjeta, que están contenidos dentro de la estructura setup, estos valores no tendrán efecto hasta que usemos esta función.

Los ejemplos de color

Este mes no hay ejemplos del CD, ya que no se ha visto ningún ejemplo real referido a lo que hemos analizado. De todas formas, nos han llegado vía e-mail quejas sobre dichos listados, pues debido a un error no aparecieron los ejemplos de algún que otro artículo. Este mes, el error será enmendado y pedimos disculpas a todas aquellas personas a las que les hayamos creado alguna molestia.

gráficos hemos eliminado realmente, pues, en el resto de aspectos, como su nombre indica, son clones.

MUSICA, MAESTRO

Un apartado de DIV Games Studio, que cumple bastante bien su cometido y del que no hemos comentado nada durante todos los artículos anteriores, es el apartado del sonido. Éste podría dividirse en dos secciones: la primera, el que se ocupa del CD Audio y la segunda, el que se encarga de producir sonidos sampleados guardados en el disco duro. Sobre el CD, señalar que podemos poner o parar cualquier canción, en cualquier momento, por medio de las funciones *play_cd()*, pasando como parámetro el número de canción, y el modo de ejecución, que varía entre tocar esa canción y seguir, o tocar y parar. Existe también la función *stop_cd()*, que detiene el aparato de forma automática. No disponemos de funciones sobre información de las pistas, pero esto no es tan necesario ya que este tipo de música sólo la utilizaremos si podemos grabar pistas de audio, y si éste es el caso, sabremos cuántas canciones tenemos. Por contra, sí podremos conocer si el CD está sonando o no, algo muy importante para que se repitan las pistas una y otra vez, usando la función *is_playing_cd()*..

No sería nada descabellado crear una tabla donde guardar todos estos identificadores de sonido, para poder tenerlos reconocidos en cualquier momento

En cuanto a los samples de sonido, son éstos los que realmente dan vida a un juego. Con este tipo de formato podremos tener almacenado cualquier sonido en el disco duro para, después, ejecutarlo. Los formatos para almacenar los samples, como comúnmente se conocen, que son leídos por DIV, son *.WAV* y *.PCM*. El primero de ellos es uno de los más extendidos en el mercado, por lo que no será difícil encontrar multitud de librerías, aparte de las que se incluyen con el producto, que tiene más de 1000 sonidos. El otro formato, *.PCM*, será el que realmente podamos usar a la hora de programar, y la calidad de que se dispone es de un sampleado a 11000 Mhz, en mono y con 8 bit. Disponemos, asimismo, de varias funciones que manejan este tipo de sonido en particular. La primera de ellas es la que nos permite cargar un sonido en memoria, que no es otra que *loas_pcm()*, y los parámetros que debemos pasar son, en primer lugar, el nombre del fichero en el disco duro y, en segundo lugar, un valor que indicara si el sonido es cíclico, es decir si se va a repetir o no, algo muy útil para cuando tengamos canciones. Esta función devuelve un valor, el identificador del fichero, que podemos usar con otra función compañera a ésta, que es *unload_pcm()*, como parámetro para descarga del fichero de sonido de memoria.
Estos identificadores de sonido serán con los que haremos referencia al sonido en memoria, y no al que está sonando, del que luego hablaremos. No sería nada descabellado crear una tabla donde guardar todos estos identificadores de sonido, para poder tenerlos reconocidos en cualquier momento. Pero vamos con el corazón de los sampleados, es decir, con la función que nos permite hacer que el sonido... ¡suene! Esta función es *sound()* y tiene tres parámetros, que son: en primer lugar, el código de fichero de sonido que nos devolvió la función *load_map()*; en segundo lugar, el volumen del sonido, y, por último, la frecuencia a que queremos que se ejecute el sonido. Estos dos últimos valores pueden variar entre 1 y 512, y en el caso de la frecuencia, cuanto mayor sea este valor más agudo será el sonido. Esta función devuelve otro identificador, que no hay que confundir con el del fichero en memoria, que denominaremos identificador del canal de sonido. Se disponen de 16 de estos canales para ejecutar sonidos, lo que nos da una buena polifonía. Si al usar la función *load_map()*, elegimos la opción de sonido cíclico, el sonido se repetirá, ocupando un canal, mientras que si ejecutamos otros 16 sonidos, se volverá a apagar. Asimismo, existe la función *stop_sound()*, que apaga el sonido, a la cual debemos pasar como parámetro el código identificador del canal de sonido que queremos que se detenga.
Otra función, muy útil en el caso de los sonidos cíclicos, es *change_sound()* que nos permite cambiar los parámetros de volumen y frecuencia que posea cualquiera de los sonidos que estén sonando. Los parámetros que debemos utilizar son, en primer lugar, el código identificador del canal que queremos modificar, en segundo, el nuevo volumen, y, por último, la nueva frecuencia. El cambiar estos parámetros puede resultarnos, igualmente, muy útil para crear *fades* de sonido, es decir, hacer que se vaya apagando poco a poco, o, si variamos la frecuencia, puede ser idóneo para simular el efecto de aceleración que tienen los ruidos de los vehículos.

Otra función, muy útil en el caso de los sonidos cíclicos, es *change_sound()* que nos permite cambiar los parámetros de volumen y frecuencia

Por ende, existen muchas funciones y estructuras para variar el volumen con carácter general, es decir, manipulan los volúmenes, al igual que las funciones anteriores, pero, en vez de hacerlo con cada sonido en particular, lo realizan manipulando los volúmenes de la tarjeta de sonido y del CD, modificando todos los sonidos, por así decirlo, a la vez. Por un lado, tenemos *reset_sound()*, que no necesita parámetros, cuya función es hacer que se reinicialice el sistema de sonido. Activará algunos de los nuevos valores que hayamos introducido en la setup, de la que luego hablaremos. Otra función de este tipo es *set_sound()*, que también activa valores de la variable setup. Esta variable la podríamos dividir en dos grupos; por un lado, los referentes a parámetros de la tarjeta, y, por otro, los que se refieren a los volúmenes generales. El primer grupo de la estructura setup la forman las variables card, port, irw, dma y dma2, de las cuales podemos encontrar más información en la documentación DIV. El otro grupo lo forman las variables master, sound_fx y cd_audio, que controlan las distintos volúmenes generales de los samplers y del cd, respectivamente. También concurre un tipo de programa especial, llamado *setup_program*, que está especializado en la modificación de estos parámetros. Todos ellos, sobre todo los de configuración de la



tarjeta de sonido, los autodetecta DIV Games Studio de forma automática cuando ejecutamos cualquier juego, por lo que estos programas quedan para los detallistas. De todas formas, podemos encontrar un ejemplo dentro de DIV Games Studio, que configura todos los parámetros de los que hemos hablado.
Pero DIV Games Studio no se queda aquí. En una futura versión profesional se está estudiando la inclusión de funciones que nos permitan manejar otros tipos de ficheros, más musicales, como pueden ser los conocidos .MOD, XM, S3M, y parecidos. Esto no se ha hecho antes por falta de tiempo, pero en un futuro, podremos usar este tipo de ficheros. Para los que los desconozcan, se trata de un tipo de ficheros con músicas, que se construyen, normalmente, con instrumentos que son samples de sonidos, es decir, podemos introducir cualquier sonido y hacer música con él. En el mercado hay multitud de programas que nos permiten construir este tipo de ficheros, pudiendo introducir nuestra propia música en DIV.

Empezaremos por definir qué es una DLL; se trata de una librería de enlace dinámico, que nos permite introducir nuevas funciones al lenguaje DIV

DLL'S Y PROGRAMACION AVANZADA

Aunque en un futuro se lleve a cabo un estudio más detallado de esta parte de la programación de DIV, vamos a realizar un somero análisis de esta parte del lenguaje. Veremos que es una dll, cómo se puede hacer, para qué sirven y su utilización, aparte de algunos trucos, para aquellos que estén empezando en este campo.
Empezaremos por definir qué es una DLL; se trata de una librería de enlace dinámico, que nos permite introducir nuevas funciones al lenguaje DIV. Estas funciones deben estar programadas en lenguajes C y ser compiladas con Watcom C, ya que se han probado con otros compiladores y no han dado los resultados correctos. Si disponemos de todas las herramientas, es decir, conocimientos de lenguajes C, el compilador Watcom en su versión 10 o superior, y ganas de programar, debemos hacer algunas modificaciones dentro del entorno de compilación para que éste cree librerías DLL compatibles con DIV, cuando se lo solicitemos. Esta modificación consiste en incluir unas líneas, dentro de uno de los ficheros de configuración de Watcom. Además, se tiene que modificar también una declaración del fichero *div.h* que, sin querer, se coló dentro del listado. Exactamente hay que cambiar la palabra *'byte *'* por la palabra *'unsigned char *'* en la definición de la función *put_sprite*, quedando algo así como:

```
__ void put_sprite(unsigned char * si, int x, int y, // Put one sprite
_________________________ int an, int al, int xg, int yg, int ang, int size, int flags);
```

Después de estas inclusiones, podremos compilar los ejemplos para ver si todo está correcto, así como observar el listado para comprobar su funcionamiento por encima. Si hacemos esto último, comprobaremos que existen distintos tipos de DLL.
Dentro de la documentación DIV, se señala que existen tres tipos de DLL: las que cogen parámetros, las de autocarga y los salvapantallas. Esta clasificación, aunque es correcta, no es del todo acertada, pues, asimismo, podría distinguirse entre las que toman parámetros y las de autocarga, considerando las de salvapantallas como una versión de las de autocarga. En esta última clasificación se crearía una nueva clase, que sería mixta, pues estaría compuesta por la unión de una dll de autocarga y otra que coge parámetros, a la vez, creando un nuevo tipo de DLL. Pero vamos a ver qué diferencian a todas estas clases de DLL.
El primer tipo comentado, las que toman parámetros, sirven para eso, y son conocidas como funciones C de toda la vida. Existe una función C, ya declarada, que sirve para este propósito, coger parámetros, debiendo retornar con otra función C declarada, que devolverá un valor de la función. Hay que señalar que los parámetros tienen forma de pila, es decir, que si mandamos cuatro parámetros en una función DIV, la función C recogerá estos parámetros al revés, el primero el último, el segundo el penúltimo y, así, sucesivamente. Como se ha comentado, siempre se debe devolver un valor, es decir, retornar un valor que recogerá la función de DIV, pero creada en C.
La segunda clase de DLL son las de autocarga, que no cogen parámetros. Estas funciones se cargan en memoria al inicio del programa DIV y se quedan residentes, hasta que el programa acabe. Son muy útiles, ya que nos permiten hacer modificaciones a bajo nivel durante todo el tiempo, mediante programación C y mientras se ejecute nuestra aplicación o juego. Por otro lado, existen una serie de funciones C, que nos brindan la posibilidad de situarnos en un paso del programa y actuar en consecuencia. Como se ha señalado anteriormente, las DLL del tipo salvapantallas pueden ser consideradas como una versión de las de autocarga, ya que se trata de funciones que son una especie de salvapantallas, es decir, se ejecutan cuando ha pasado determinado tiempo. También existen variables y funciones C predefinidas que se encargan de tomar estos valores, sobre tiempo y acciones sobre el salvapantallas, que podremos modificar según nuestros gustos.

Como se ha señalado anteriormente, las DLL del tipo salvapantallas pueden ser consideradas como una versión de las de autocarga

Pero puede darse el caso de que necesitemos coger parámetros de una función, desde DIV, y a la vez querer que nuestra función se autocargue, es decir, que esté todo el tiempo en memoria para que ejecute una serie de acciones en la pantalla. Entonces, es cuando entran en juego los tipos de DLL mixtas que conjugan las características de los otros dos tipos para crear uno nuevo, que no nos dejará ninguna limitación a la hora de programar cualquier cosa en C para nuestros juegos DIV.
Como se ha comentado, en próximos artículos se verán más detenidamente todas las funciones predefinidas C que se encuentran en *div.h*. Además, realizaremos alguna DLL para ver cómo se realiza.

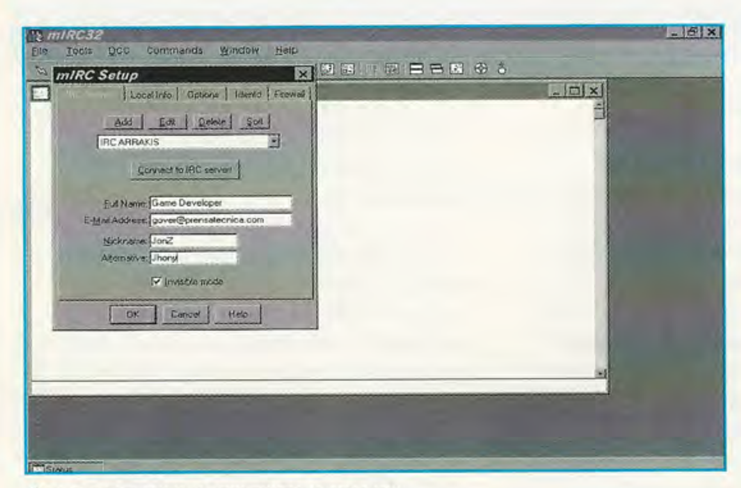
Conclusión

Hemos visto aspectos acerca del ratón, aspectos sobre la música, sobre los clones y sobre la DLL, de ahí que esperamos que los temas tratados hayan sido los correctos y que no queden dudas a nadie. Si ha sido así, podéis mandar un e-mail a tizo@100mbps.es donde intentaremos solucinarla. Asimismo, señalar que ya se ha actualizado la Web de DIV, con los resultados del concurso. Hasta el mes que viene y que os DIVirtais programando.

Autor: Leticia Krahe

Chat en la red

Seguro que sois muchos los que conocéis las prestaciones de algunos de los programas que circulan por la red, pero seguramente no conoceréis las nuevas prestaciones que algunos de estos programas de conexión y navegación ofrecen al usuario.



ESTA ES LA VENTANA DE CONEXION AL SERVIDOR.

En este artículo vamos a hablar de aquellas posibilidades que ofrece un módem, una conexión a Internet y algunos programas shareware que amplían las capacidades de los usuarios que sólo funcionan con NetScape o Explorer.
Un fenómeno que surgió casi a la vez que la propia explosión de Internet fue la creación o rápida difusión de los canales de charla, más coloquialmente llamados IRC (*Interactive Real Chat*).

PRECEDENTES

Estos canales, mantenidos por un servidor con este servicio, ofrecen al usuario de Internet grandes posibilidades de comunicación en tiempo real con más usuarios de la red.
Como ejemplo, podemos hablar de conectarnos al servidor *Hispano.org*, por señalar alguno, por Arrakis, mismamente, usando el programa de mayor difusión para este tipo de conexión, sin lugar a dudas, el MIRC52, un programa shareware que podremos encontrar en infinidad de servidores o incluso en el CD que acompaña esta revista.
Dicho programa nos permite conectarnos, previa correcta configuración, a más de quinientos canales de charla. En *Hispano.org* se recogen muchos canales mantenidos por una gran cantidad de servidores de habla hispana, bien en España o fuera de nuestras fronteras. Una vez configurado el MIRC, únicamente habrá que configurar la dirección IP, al teléfono al que se llama, el canal habilitado por el servidor para la conexión, que suele ser en el caso de Hispano.org y Arrakis, del 6665 al 6668, nuestro "nickname" o apodo, nuestra dirección de correo real, un "nickname" de repuesto, y el nodo en el que queremos conectar.

AQUI PODEIS VER ALGUNOS DE LOS CANALES DISPONIBLES.



CONECTADOS

Una vez conectados, lo primero que deberemos hacer es coger la lista de canales disponibles, hecho para el cual tendremos que indicar cuántas personas, como mínimo, deseamos que haya en el canal (si sólo hay dos o tres personas, no estará muy animado).
De hecho, todos los canales habituales están mantenidos por lo que se denomina un "robot", que nunca se va del canal y es el que lo mantiene hábil todo el tiempo. En determinados casos, estos robots están programados para no tolerar insultos o palabras malsonantes, (es raro que un robot nos eche del canal). Por lo general, estos robots no son más que scripts en C que están hechos para que determinadas personas, normalmente las habituales del canal, sepan cómo conseguir privilegios y cosas así.
Dentro de IRC, y una vez que llevemos conectados un rato, veremos que se trata de otro mundo, pues todo tiene su nombre particular, su "jerga"... Así pues, echar a alguien del canal se denomina "*nuckear*", si alguien se desconecta por culpa de la conexión, "*se cae*", escribir en mayúsculas es gritar... todo tiene su código.
Podremos ver a todos y cada uno de los usuarios que están conectados en ese momento y, o bien charlar en el canal, o bien pasar a uno privado pulsando dos veces sobre su nombre en la lista de usuarios de la derecha.
Mediante este tipo de conexión también se pueden bajar datos que son enviados por otros usuarios, vía DCC, lo cual facilita mucho las cosas, poder hablar al mismo tiempo que bajas ficheros o envías ficheros.

COMANDOS

Existe un sinfín de comandos para llegar a controlar bien el MIRC, pero digamos que algunos de ellos, para que os hagáis una idea, son */WHOIS* seguido del nombre; esto nos dirá quién es realmente o, al menos, lo intentará, el usuario real al que le corresponde el "nick" que siga al "whois".
Otro de los comandos es */ME*, seguido de una frase; esto hará que "/ME tira un tomate a PIRRO", de como resultado "Leticia tira un tomate a PIRRO".
Se pueden escribir frases de diversos colores, muy fácilmente, pulsando CTRL + K, y después el número de tinta que queremos utilizar, desde la 0 a la 9, siendo, por ejemplo, el 4 la roja.

EN DEFINITIVA

La IRC ofrece nuevas posibilidades para disfrutar más aún de la red mediante programas como MIRC. Su última versión se puede conseguir fácilmente en miles de lugares y funciona a la perfección. Bueno chicos, nos vemos en el chat.

Autor: Juan Suárez

El diseño de niveles [y III]

Proseguimos con la andadura de esta exclusiva sección, que os propone llegar a conocer, poco a poco, los entresijos del desarrollo de un videojuego. Podréis llegar a descubrir algunos de los aspectos más recónditos de un desarrollo y conocer todas las piezas de un complejo puzzle.

Este mes, y tal y como os habíamos adelantado en el anterior artículo, hablaremos de cómo organizar vuestro material, para que os lancéis a la búsqueda de un puesto de trabajo en una empresa de desarrollo de videojuegos. Como comentábamos también en la última entrega de esta colección de artículos, os aconsejamos que vayáis desarrollando un "pellejo" duro y resistente, porque la tarea de encontrar un hueco en esta industria no es siempre fácil. Y aunque recibamos muchas negativas al principio, no hay que desanimarse, porque si nuestro trabajo es realmente bueno, habrá siempre alguien que sepa apreciarlo.

ALGUNOS CONSEJOS PRACTICOS

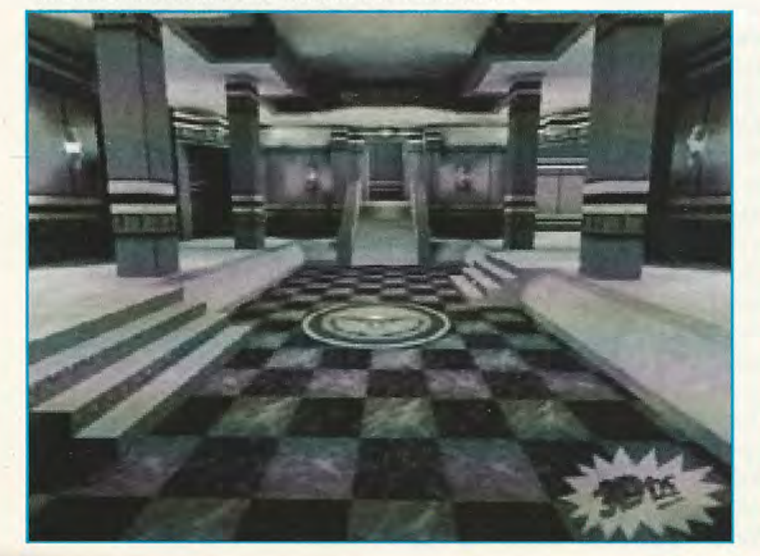
Nuestro primer consejo es que, antes de que os lancéis a la desesperada en busca de un trabajo, desarrolléis previamente una basta experiencia demostrable. Es decir, que hagáis tantos niveles como podáis. Conseguid soltura en el desarrollo de niveles. Observad los niveles hechos por otras personas y escoged aquellos que más os gusten para, después, recrearlos por vosotros mismos. Intentad transmitir el mismo "feeling" que percibisteis al probar aquellos niveles. Procurad comprender qué es lo que os resulta atractivo de esos niveles que habéis tomado como ejemplo: el tamaño, las proporciones, la iluminación, las texturas, la geometría con que están decorados, etc... Una vez que hayáis entendido qué es lo que hace a esos niveles algo atractivo, divertido, o cualquiera que sea el sentimiento especial que os han transmitido, intentad aplicar esas mismas ideas a vuestro propio diseño. Emplead todo el tiempo necesario para recrear una iluminación convincente porque éste es, sin duda alguna, el aspecto más importante en la ambientación de un nivel. Una buena iluminación dirá mucho en vuestro favor, aunque la geometría sea mediocre. Evidentemente, este último aspecto es también muy importante. Pero son muchos los niveles que pasan desapercibidos, pese a sus sorprendentes diseños arquitectónicos, por culpa de una mala iluminación. Así que lo mejor es, como era de esperar, conseguir ambas cosas. Pero no queremos dejar de destacar que la iluminación es la verdadera clave para recrear un nivel convincente y realista.

Si nuestro trabajo es realmente bueno, habrá siempre alguien que sepa apreciarlo

METODOLOGIA A SEGUIR

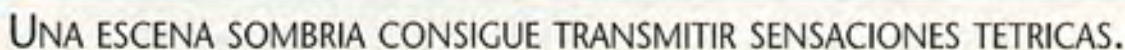
Para llegar a ser un diseñador de niveles aventajado conviene conocer unos cuantos "truquillos" que, sin ser realmente reveladores para nadie, dada su obvia lógica, pueden pasar inadvertidos a mucha gente. Estos "truquillos" os ayudarán enormemente a haceros notar y, en definitiva, conforman una metodología a seguir previa muestra de vuestro trabajo a una compañía desarrolladora. Emplead cualquier herramienta de creación tridimensional que esté a vuestro alcance, ya sea 3D Studio, LightWave, WorldCraft, o cualquier otra con la que os sintáis cómodos diseñando. En este sentido os aconsejamos que empleéis, si es posible, un editor que aproveche las características de un juego ya desarrollado como pueda ser Quake o UnReal. De esta forma, estaréis mostrando todo vuestro potencial como diseñadores, permitiendo a la persona que juzgue vuestro trabajo jugar a un nivel diseñado por vosotros mismos, poniendo de manifiesto no sólo vuestras dotes de diseñador, sino también las de gran jugador. Si esto os resulta demasiado pedir, simplemente crear una animación simulando jugar una partida, haciendo que los supuestos elementos de interacción creados

EJEMPLO DE UNA COMPOSICION ARQUITECTONICA EXQUISITA.



EL RECORRIDO DE UN MAPA PUEDE SIMULAR GRANDES ESTANCIAS, AUNQUE ESTAS NO LO SEAN.



UNA ESCENA SOMBRIA CONSIGUE TRANSMITIR SENSACIONES TETRICAS.



UNA CORRECTA ILUMINACION LOGRA AUMENTAR EL REALISMO DE LA ESCENA.

en el nivel se activen tal y como ocurriría en el supuesto juego.
Bueno, sin más dilaciones vamos con los consabidos consejos que definen la metodología a seguir :

1. Primeramente, y como ya hemos dicho cientos de veces, echad un detallado vistazo a algunos de los mejores diseños de niveles hechos en la historia del videojuego, como los de Quake I y II, Duke Nukem 3D, o incluso DOOM. Podréis apreciar que todos ellos están pensados para que, aparte de ser tremendamente interactivos, aprovechen todo el espacio de forma que se puedan recorrer "de cabo a rabo". Si sois observadores, podréis ver también que se simulan mapas de grandes dimensiones, cuando a decir verdad el espacio total de los mapas es bastante reducido. Esto se consigue llevando al jugador por zonas que, en realidad, son contiguas a otras que parecen estar muy alejadas de las primeras, o haciendo pasar varias veces al jugador por una misma sala, a la que se llega desde varios sitios. En nuestra opinión, un claro ejemplo podría ser el nivel E1M2 de Quake que, además, cuenta con una estupenda iluminación y una composición arquitectónica sorprendente, por no mencionar la enorme jugabilidad. También cuenta con algunos "golpes de efecto", como cuando, por ejemplo, se llega al final del mapa y nos encontramos con unos cubos que levitan del suelo, que se ven seguidos de la retirada de la viga que obstruye la puerta de llegada. Entonces ésta se abre, dejando aparecer a un par de demonios que se abalanzan sobre nosotros, seguidos de unos cuantos enemigos más. Este truco hizo que más de uno se cayese de la silla, o que necesitase otro par de calzoncillos limpios... Es una idea simple, pero demoledora...
2. Cread unos pocos niveles excepcionales y no un montón de buenos niveles. Dos o tres mapas excelentes son mejores que veinte decentes. Intentad que esos niveles incorporen algo único. Usad vuestra creatividad para crear algo que no hayáis visto nunca en ningún otro nivel y procurad construir una arquitectura convincente y espectacular, prestando especial atención a detalles como el correcto "tileado" de las texturas, consiguiendo que la iluminación sea perfecta. Esto puede llevar días, no horas... Así que tomároslo con calma.

Dos o tres mapas excelentes son mejores que veinte decentes

3. Testad y depurad vuestros niveles. Jugad a ellos en modo individual (*single player*) y haced que vuestros amigos jueguen también, pidiéndoles después un amplio *feedback*. Jugad a ellos en modo *multiplayer* hasta la saciedad. Haced un balance perfecto de jugabilidad, y ajustad el nivel de dificultad, colocando adecuadamente los *ítems*, las armas, los enemigos, etc... Enseñádselos a vuestra madre y a vuestro padre, y prometedles que sólo es una etapa de vuestro desarrollo, que pronto maduraréis... En serio, testad vuestros niveles hasta que los odiéis...
4. Cuando creáis haber logrado algo realmente bueno, entonces estaréis listos para enseñarlo. Haced un *upload* a través de la red a todos los sitios que penséis pueda interesar vuestro trabajo, procurando incluir un fichero de texto con detalles sobre el trabajo enviado, así como vuestros datos personales, incluido el e-mail, por supuesto. Buscad en la WEB todas las compañías desarrolladoras a las cuales penséis que puede interesar vuestro trabajo. Muchas de ellas ofrecen en sus páginas ofertas de trabajo, entre ellas las de diseñadores de niveles. Intentad contactar directamente con ellas y ofrecedles vuestros diseños. No subestiméis vuestro trabajo, ni os vendáis baratos... Mostrad confianza en vosotros mismos y decidles que sois magníficos, grandiosos..., convenciéndoles para que admitan echar un vistazo a vuestros increíbles niveles. Si realmente lo son, sabrán apreciarlo.

HASTA AQUI HEMOS LLEGADO...

Bueno, concluye la sección "Diseño de Niveles" que esperamos os haya sido de interés y provecho. Pero no queremos despedirnos sin antes daros unos últimos consejos. Procurad aprender todo lo posible sobre cómo funcionan las "engines", o motores 3D, de los últimos juegos que salen al mercado. Haceros con el manejo de todas las herramientas de creación tridimensional posibles, así como editores de niveles y programas de creación bidimensional (Photoshop, Fractal Painter, etc...). Haced cuantos mapas os sea posible, y aprended a ver cada centímetro de detalle en vuestra cabeza antes de poneros a diseñar un nivel. Imagináos a vosotros mismos como encargados del diseño de niveles de un nuevo juego que está desarrollando una empresa. Ponéos manos a la obra y haced de ese juego el más espectacular y divertido de cuantos se hayan podido crear en la industria... Si os va el tema, tomároslo en serio, porque seguro que si insistís llegaréis a ser unos grandes diseñadores. Y pese a todo, recordad que el camino puede ser duro, pero no os desaniméis ni perdáis la ilusión. Eso es siempre lo último... Os deseamos mucha suerte en vuestra andadura como diseñadores y el próximo mes os seguiremos desvelando más entresijos de este apasionante mundo del desarrollo de videojuegos. N'xaludito... y hasta el mes que viene...

Autor: Víctor Segura

Voces en un videojuego

Este mes nos introduciremos en el mundo de la grabación y el tratamiento de la voz analizando los diferentes pasos que debemos seguir para grabar correctamente las voces e incluirlas en nuestro propio videojuego.

Una vez terminado un videojuego, los diferentes departamentos de desarrollo (programación, gráficos y sonido) se encargan de "depurar" y repasar minuciosamente su trabajo para obtener así el mejor resultado posible. En cuanto a sonido se refiere, se evalúa si, efectivamente, las decisiones tomadas en la banda sonora (estilo de música empleado y distribución de las canciones a lo largo del videojuego) son las correctas y si los efectos de sonido utilizados reflejan el realismo pretendido desde un principio. En muchas ocasiones, el resultado es más que satisfactorio; aun así conviene plantearse el hecho de introducir voces que acompañen al resto de los sonidos para dotar al videojuego de un mayor dinamismo y, por qué no, más frescura y diversión en determinados momentos del juego.

¿DONDE INTRODUCIR LAS VOCES?

Hay que tener mucho cuidado a la hora de introducir las voces en el videojuego, ya que es importante tener en cuenta tanto el lugar en el que se van a introducir como la cantidad de voces disponibles. Lo primero que debemos hacer antes de grabar las voces es estudiar cuidadosamente las voces que vamos a necesitar, el estilo que queremos dar a dichas voces y los medios disponibles para grabar, procesar y modificar las voces.

Por lo general, un videojuego suele tener tres partes: la intro, los menús y las fases. Dependiendo del tipo de videojuego que estemos desarrollando incluiremos las voces en las diferentes partes. Normalmente, las voces que aparecen en la intro suelen ser o voz en off, con subtítulos en pantalla, o la voz de los diálogos de los protagonistas de dicha intro. Durante los menús suelen ser voces que van relacionadas con los protagonistas o que indican o repiten las diversas acciones que realizamos a lo largo de las pantallas de los menús. Y durante las fases pueden ser voces de los personajes del juego que contestan a órdenes que les damos o, por ejemplo, en el caso de juegos deportivos, comentarios dinámicos simulando los de los comentaristas deportivos.

Lo primero que debemos hacer es estudiar cuidadosamente las voces que vamos a necesitar

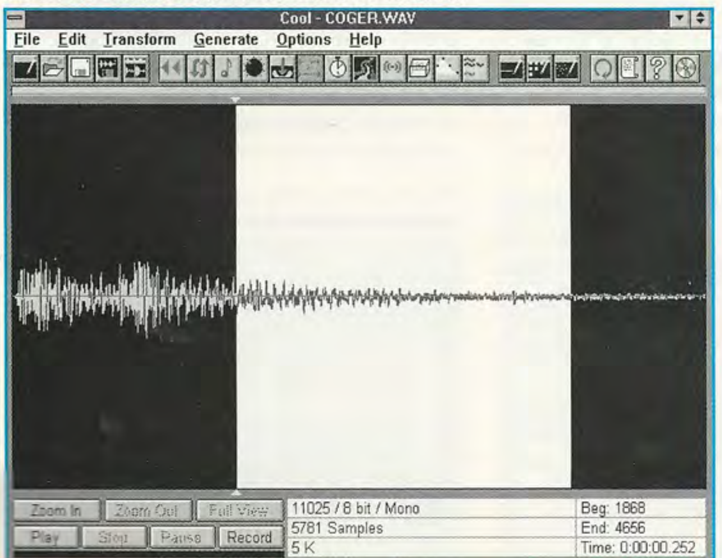
ESTUDIO DE ESTILO DE VOZ

El hecho de acertar en el estilo de las voces puede ser la clave del éxito sonoro en el videojuego, recordemos el DukeNukem 3D, donde las voces del protagonista ayudaban a que el juego fuera más divertido. Sin embargo, en juegos como el Battle Arena Toshindden la voz que narra la historia de la acción del juego consigue introducir al jugador en una atmósfera diferente, en la que el único objetivo es derrotar al resto de los adversarios o morir honorablemente. Éstos son dos ejemplos muy diferentes de utilización de voces en un videojuego, pero que cada uno, en su caso, consigue motivar al jugador y lograr así el objetivo deseado con la inclusión de las voces. También es importante analizar el género de los sonidos que vamos a utilizar, es decir, voces masculinas o femeninas.

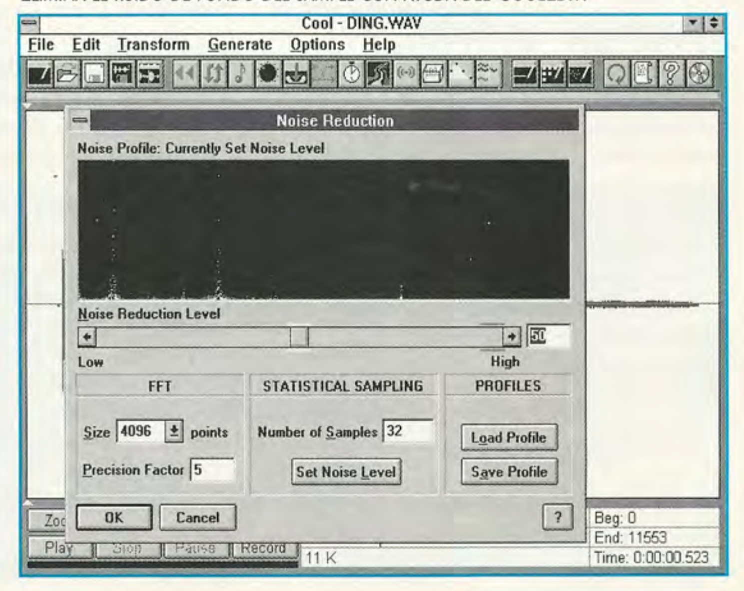
SATURACION

Como cualquier trabajo relacionado con los efectos de sonido tenemos que tener mucho cuidado con no saturar de sonidos el videojuego. Por ejemplo: supongamos que queremos incluir voces en una intro de un videojuego de estrategia. Podemos optar por dos opciones totalmente diferentes. Una

GRABA TUS VOCES DENTRO DE LOS LIMITES DE VOLUMEN.



ELIMINA EL RUIDO DE FONDO DEL SAMPLE CON AYUDA DEL COOLEDIT.



Grabación de voces paso a paso

Para grabar las voces de un videojuego y, en general, de cualquier aplicación informática es conveniente seguir estos pasos:

1.- Probar que la voz no llega al valor máximo del volumen de grabación. Si alcanzara dicho valor, bajar el volumen de entrada del micrófono.
2.- Grabar la voz al ordenador sin ningún tipo de efecto. Procurar que la voz sea lo más limpia posible.
3.- Pasar un filtro de reducción de ruido.
4.- Aplicar los efectos necesarios.

posible opción es la de utilizar las voces para narrar la historia y los acontecimientos precedentes a la acción del videojuego acompañando a la sucesión de imágenes de dichos acontecimientos, y, la otra, incluir determinadas voces tipo diálogos de los propios protagonistas de los acontecimientos. Elegir una de estas dos opciones y su posterior realización puede ser igualmente satisfactorio. Pero podemos pensar también en mezclar la narración de los acontecimientos por un lado, junto con los diferentes diálogos de los personajes de la Intro. Sin duda alguna, es muy posible que esta última elección pueda saturar al jugador, pues hay que tener en cuenta que lo que pretendemos con el uso de las voces es mejorar el resultado final de nuestro trabajo, pero hemos de tener cuidado y no sobrecargar con sonidos súperespectaculares el videojuego.

En la música la calidad no siempre va relacionada con el dinero, pero en el caso de los micrófonos sí sucede así

MICROFONOS Y SOFTWARE

Aunque el típico micrófono de sobremesa que viene con las tarjetas de sonido puede ser suficiente para grabar nuestras voces, conviene invertir alrededor de las 15.000 pesetas en un micrófono con intenciones (aunque sólo sea eso) más profesionales. En la música la calidad no siempre va relacionada con el dinero, pero en el caso de los micrófonos si sucede así, y si pensáis comprar un micrófono de 3.000 o 4.000 pesetas, mejor es que utilicéis el que viene con la tarjeta de sonido. Cualquier programa de edición de FX valdrá para proporcionar a nuestras voces una calidad más que suficiente.
A continuación, mostramos el proceso que debemos seguir para la grabación de una voz suponiendo que todo el equipo del que disponemos es un ordenador, una tarjeta de sonido y el software de edición de sonidos.

GRABACION INICIAL

Antes de grabar tenemos que asegurarnos de conectar el micrófono en la entrada MIC y no en LINE IN. Si al grabar encontramos mucho ruido de fondo puede ser que hayamos conectado el micro en LINE en vez de en la entrada MIC. Una vez aseguradas las conexiones conviene desactivar los controladores de entrada de sonido que no se utilicen en el momento de la grabación (LINE IN), ya que podrían introducir señal de ruido que disminuya la calidad de sonido del sample que vamos a grabar.
Antes de grabar tenéis que aseguraros de que no sobrepasáis los niveles de entrada de volumen, es decir, si trabajáis con el soundforge evitar que la señal que indica el volumen de la voz se ponga de color roja. En el caso de que no tuvierais un indicador de volumen de grabación tenéis que realizar una grabación de prueba y ver que el dibujo del sample no *choca* con los límites de la pantalla, o sea, la altura máxima y la mínima del sample no toca con el borde de la pantalla donde está dibujado.

ELIMINACION DE RUIDO

Una vez grabado el sample, y antes de aplicarle cualquier efecto, tendremos que aplicarle una función de análisis de ruido, y si tiene ruido habrá que pasarle el efecto de *noisereduction.*
La mayoría de los programas de edición de audio (Cool Edit, SoundForge, etc...) tienen una opción que calcula el índice de ruido de la parte seleccionada de un sample. Para calcularlo debemos seleccionar una parte del sample que no tenga sonido, que sea silencio, ya que el ruido se aprecia más en zonas silenciosas que en zonas en las que haya sonido.

ECUALIZACION Y DEMAS EFECTOS

Cuando hemos dejado nuestro sample limpio (bien grabado, recortado y sin ruido) es cuando procedemos a aplicarle los diferentes efectos necesarios para crear determinadas atmósferas.
Tenemos que ser cuidadosos y no aplicar "todos" los efectos al sample, sino que conviene que sean variaciones sutiles del sample original. Lo primero que tenemos que conseguir es una ecualización correcta. Una vez ecualizado, le podemos aplicar los diferentes efectos. Probablemente tengamos que aplicarle algún tipo de Reverb, efecto que consigue muy buenos resultados aplicado a voces. Luego, dependiendo del gusto de cada uno, se le puede aplicar algún flanger para dar un aspecto más futurista o cualquier efecto que satisfaga la intención del músico.

Tenemos que ser cuidadosos y no aplicar "todos" los efectos al sample, sino que conviene que sean variaciones sutiles del sample original

CALIDAD

Claro está que la calidad óptima de grabación sería calidad CD, es decir, 16 Bits, estéreo y a 44100 Hz, pero hemos de considerar que a mayor calidad más memoria ocupa. Así pues, una calidad más que suficiente que sea un término medio entre buen sonido y poca memoria sería 16 bits, 22000 hz y en mono.

UN ULTIMO CONSEJO

En muchas ocasiones habréis visto que los reporteros de televisión y programas de radio tienen el micrófono rodeado de gomaespuma. Esto es porque, a veces, las eses se marcan demasiado o las pes provocan un pico de graves. Si podéis comprar el micrófono con la *alcachofa* evitaréis horas de ordenador tratando de corregir problemas con las eses y las pes que podríais haber evitado de antemano.

PASAR UN FILTRO DE REDUCCION DE RUIDO.



Autor: Víctor Segura

CAKEWALK [y III]

Terminamos con el curso de Cakewalk explicando la manera de configurar y gestionar los puertos MIDI. Asimismo, veremos cómo se configuran los diferentes canales asignándoles distintos instrumentos y sus correspondientes sonidos.

Para terminar con el curso de introducción al Cakewalk vamos a comentar el *Settings menu* y el *Tracks menu*. Tenemos que tener en cuenta que todas las opciones que comentemos serán de gran ayuda, sea cual sea el secuenciador que usemos, ya que la base de todos los secuenciadores MIDI es la misma. El hecho es que mediante lo que hemos aprendido a lo largo del curso, vamos a saber utilizar con cierta facilidad nuestro secuenciador y, además, tendremos la base suficiente para enfrentarnos a otros secuenciadores y poder "juguetear" con ellos.

SETTINGS MENU

En este menú es donde gestionamos la configuración del secuenciador, ya sea configuración de drivers MIDI, la asignación de bancos de sonidos a los diferentes canales o la gestión del MIDI OUT,THRU e IN. A continuación, introducimos los diferentes comandos que forman el *Settings menu.*

• **Metrónomo**

Es un instrumento de medición de compases. En la práctica se utiliza (sobre todo en clases de música) para que el alumno o el músico lleve el ritmo musical correctamente, sin anticiparse o retrasarse. El metrónomo que incluye el secuenciador tiene la misma utilidad, pero cuenta con unas determinadas ventajas sobre los metrónomos convencionales, ya que podemos configurarlo para que realice diferentes clases de marcado de ritmo. Para acceder a esta utilidad lo haremos mediante la barra de menús *Settings>Metronome.*

Aparecerá una ventana con diversas opciones divididas en tres partes diferentes: la primera (general) es donde configuramos los diferentes momentos en los que queremos que suene y en qué tiempo, es decir, si queremos que suene durante la grabación, durante el playback de la canción y si queremos que acentúe el tiempo golpe de cada compás con más intensidad que el resto de los tiempos. También existe una casilla, llamada Count-in, con la que configuramos cada cuánto tiempo (tiempo de compás) ha de sonar el metrónomo. La segunda (MIDI note) indica la nota que emulará el sonido del metrónomo que, generalmente, suele ser F#3 del canal 10, y su duración, que suele ser de un tiempo. La última es la opción que permite que la salida del sonido sea el PC speaker.

• **Timebase**

Mediante Settings>Timebase... accedemos a la opción timebase, que indica el tiempo base de nuestra canción. Este tiempo se mide por pulsaciones por minuto, es decir, si configuramos el tiempo base a 120, equivaldría a 120 pulsaciones por minuto (dos tiempos por segundo).

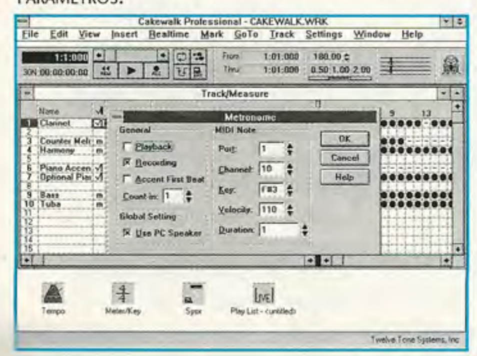
• **MIDI Devices**

Por lo general, el problema al que nos enfrentamos cuando instalamos un programa nuevo en el ordenador y, en particular, si es un programa de audio, es la configuración de los puertos de entrada y salida del sonido. Claro está que la configuración depende del equipo que tengamos, es decir, si los sonidos los "cogemos" directamente de la tarjeta de sonido o, por el contrario, los "cogemos" de una fuente externa como puede ser un módulo de sonido o un teclado. Para las personas cuya fuente de sonidos sea la propia tarjeta de sonidos, en la parte correspondiente a OUTPUT deberá marcar el controlador de su tarjeta (específico a cada tarjeta). Por ejemplo, los usuarios de la SB16 deberán marcar el *Voyetra Super FM driver.* Si se tiene otra tarjeta, hay que marcar el que no es ni SB16 MIDI OUT ni Mapeador MIDI microsoft. Si la fuente de sonidos es un módulo de sonidos o cualquier aparato externo que está conectado vía MIDI, la opción que se debe elegir es SB16 MIDI OUT. Para el puerto de entrada INPUT, se puede elegir el TTS VIRTUAL PIANO, si se toca desde el teclado del ordenador, o el SB16 MIDI IN, si se toca desde un teclado maestro o un sintetizador. Hay que tener en cuenta que cada tarjeta trae sus propios controladores, por ello, no será nada sorprendente que cuando suena el teclado no suena la canción, o viceversa. Si esto sucede, el problema está precisamente en la configuración de los puertos de entrada y salida de sonido, es decir, en Settings>MIDIDevices.

MIDI IN, MIDI THRU Y MIDI OUT

Desde el Menu Settings podemos acceder a estas tres opciones de configuración de los puertos MIDI. La configuración de dichos

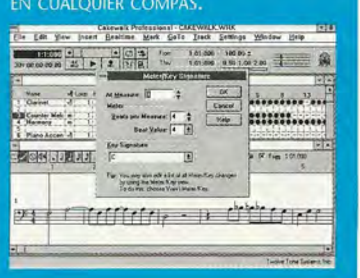
VENTANA DEL METRONOMO CON SUS DIFERENTES PARAMETROS.



Cambios de tiempo y armadura

Una de las muchas ventajas de los sistemas MIDI es que se puede modificar en cualquier compás de nuestra canción el tiempo y la clave. Podemos hacer que del primer compás al vigésimo primero estemos en DO Mayor y que del vigésimo segundo en adelante estemos en Mi bemol Mayor, además se puede modificar también del cuatro por cuatro inicial a cualquier otro compás como puede ser un seis por ocho. De hecho, tenemos la posibilidad de editar la lista de cambios utilizando la opción View>Meter/Key, y ver todos los cambios de tiempo que hemos realizado a lo largo de la canción.

CAMBIA EL TIEMPO Y/O LA ARMADURA EN CUALQUIER COMPAS.



Definición de instrumentos

Cakewalk nos permite especificar una "definición de instrumentos" para configurar cada puerto y canal en función de los instrumentos disponibles. Para acceder a esta opción lo haremos mediante comando de menús Settings>Instruments. La definición de instrumentos informa al Cakewalk sobre determinados aspectos del sintetizador como:

- **Qué nombres se usan para los sonidos, notas y controladores.**
- **Dónde contiene sonidos de baterías un instrumento.**
- **Cuál es el método de selección de bancos que usa el sintetizador.**

La lista de instrumentos es la siguiente:

Alesis D4	Alesis HR-16 Drum Machine
Alesis Midiverb	Alesis Quadraverb Plus
ART ProVerb 200	Boss SE-50
Casio CT-470	Creative Labs WaveBlaster
Creative Labs Wave Blaster Drum	E-Mu Proteus
Ensoniq Ks-32	Ensoniq Ts-10
Ensoniq VFX	General MIDI
General MIDI Drums	Kawai Gmega
Kawai k-11	Kawai k-11 Drums
Korg 01/w	Korg M1
Kurzweil*	Lexicon*
Mackie OTTO-1604	MOTU Midi Mixer 7s
Roland GR*	Roland GS*
Roland JV*	Roland MT*
Technics	Yamaha

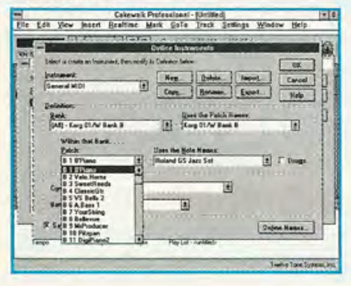
* todos los modelos pertenecientes a ese instrumento

En el caso de que nuestro módulo o sintetizador no aparezca en esta lista, podemos definirlo mediante la opción Define Instrument.

puertos dependerá, entre otras cosas, del equipo que tengamos y de su distribución. Para configurar estos puertos se necesitan unos conocimientos MIDI un poco más avanzados (como la sincronización de eventos MIDI, por ejemplo), razón por la cual trataremos este punto más adelante cuando hablemos de la sincronización MIDI y de muchas otras tareas relacionadas con este tema.

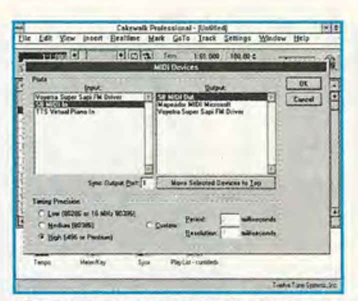
• Record Filter

Cuando estamos grabando una melodía desde un teclado, podemos grabar no sólo las notas que estamos tocando sino también los cambios de instrumento que hagamos a tiempo real, las variaciones del Pitch Wheel, el aftertouch y los cambios de los valores de los diferentes controladores que podamos modificar directamente desde teclado. Pues bien, para poder configurar todas estas opciones que queremos activar y/o desactivar durante la grabación, lo haremos desde la opción *Settings>Record Filter* seleccionando (en el caso que sea necesario) las opciones descritas. Todos estos cambios, que gracias al *Record Filter* podemos realizar a tiempo real durante la grabación, se pueden llevar a cabo a posteriori, como comentamos en el primer capítulo de este curso, modificando los distintos controladores.


DEFINE LOS NOMBRES DE TUS BANCOS DE SONIDOS.

• Configuración Canal-Instrumento

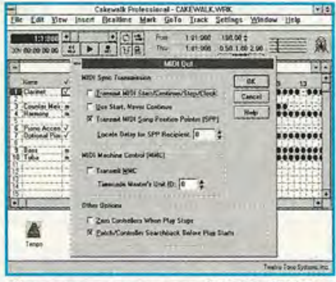
Al comenzar una canción por defecto, seleccionemos el canal que seleccionemos, tenemos asignados los instrumentos pertenecientes al banco de sonidos General MIDI. Si tenemos una tarjeta de sonido, un módulo o un sintetizador con más de un banco de sonidos al que accedemos (por ejemplo) poniendo los controladores 00 y 32 al valor cero, sonarán sonidos de esos otros bancos de sonido, pero aparecerán con los nombres del banco GeneralMidi. Mediante la opción *Settings>Instrument* podemos asignar diferentes nombres de diversos equipos con sus bancos de sonidos a los canales de nuestra tarjeta. Por ejemplo, si tenemos por costumbre utilizar en el canal 1 el sonido de piano (sonido número 21) que pertenece al banco A de nuestro módulo de sonidos y que es mejor que el sonido de piano del banco GM, deberíamos configurar el canal 1 con su *Uses Instrument* correspondiente para que, al estar en el canal 1, salgan los nombres de sonidos correspondientes a nuestro banco de sonidos, es decir, que el sonido 21 tenga como nombre PIANO y no Accordion, correspondiente al sonido número 21 del banco GM. Para poder configurar los instrumentos de las diferentes fuentes de sonidos tenemos que pulsar el botón define, seleccionar el instrumento y su banco correspondiente. En el caso de que nuestro módulo o sintetizador no apareciera en la lista de instrumentos, podemos definirlo junto a sus bancos y sonidos correspondientes.


VENTANA DE CONFIGURACION DE LOS PUERTOS DE ENTRADA Y SALIDA.

• Track Menu

Por último, vamos a ver las diferentes opciones del Tracks menu. Estas opciones son:

- *Parameter*: Opción con la que podemos acceder a los diferentes parámetros asociados a una pista, como son: *Name, status, archie, loop, vel, key, time, port, channel, bank, patch, volume, pan.*
- *Solo*: Si tenemos seleccionada una pista y accedemos a esta opción, quedará sonando sólo esta pista.
- *Un-Solo*: Para que se oigan el resto de las pistas (funciona si, anteriormente, se ha usado la opción Solo).
- *Clone*: Clona una pista sobre otra, es decir, hace una copia de una pista en otra.
- *Wipe*: Borra todos los eventos, pero no resetea los parámetros de dicha pista.
- *Kill*: Borra todos los eventos y, aparte, resetea todos los parámetros.
- *Sort*: Ordena las diferentes pistas. Puede ordenar entre otros por nombre, por puerto, por canal, por seleccionados... Se puede ordenar ascendente o descendentemente.


CONFIGURACION DEL MIDI OUT A TRAVÉS DEL CAKEWALK.

PROBAR Y PRACTICAR

Aunque, a veces, resulta un poco pesado meterse a fondo con un programa de ordenador, ya sea de sonido o no, durante estos tres meses hemos dado las nociones básicas de los secuenciadores, nociones suficientes como para enfrentarnos a cualquier secuenciador y poder realizar las canciones que formarán la banda sonora de nuestro videojuego. Desde aquí recomendamos que se "juegue" con el programa probando las diferentes opciones de grabación, de control de eventos, etc, ya que, en muchas ocasiones, el factor suerte nos ayuda a descubrir nuevas opciones que pueden ser en algunos casos de gran utilidad.

